composer2d.py

```python
from utils import *
from shape2d import Vertex, Shape2D

def Line(_position : Vertex = Vertex(0, 0, 0), _edge : float = 1, _color : tuple[float] = None, _rotation : tuple[float, str] = None):
    """
    Initialize 2D Line object .
    """
    half_edge = _edge / 2
    VERTICES = [
        Vertex(
            _position.x,
            _position.y,
            _position.z - half_edge,
        ),
        Vertex(
            _position.x,
            _position.y,
            _position.z + half_edge,
        ),

    ]
    return Shape2D(_position, VERTICES, _color, GL_LINES)


def Axises():
    """
    Initialize 2D Axises object .
    """
    half_edge = 4
    VERTICES = [
        Vertex(
            0,
            0,
            0,
        ),
        Vertex(
            0,
            0,
            -half_edge,
        ),

        Vertex(
            0,
            0,
            0,
        ),
        Vertex(
            0,
            0,
            half_edge,
        ),

        Vertex(
            0,
            0,
            0,
        ),
        Vertex(
            0,
            -half_edge,
            0,
        ),

        Vertex(
            0,
            0,
            0,
        ),
        Vertex(
            0,
            half_edge,
            0,
        ),

        Vertex(
            0,
            0,
            0,
        ),
        Vertex(
            -half_edge,
            0,
            0,
        ),

        Vertex(
            0,
            0,
            0,
        ),
        Vertex(
            half_edge,
            0,
            0,
        ),

    ]
    return Shape2D(Vertex(0, 0, 0), VERTICES, COLORS['green'], GL_LINES)


def Net(_size : int = 4):
    """
    Initialize 2D Net (every 1 unit = line) object .
    """
    len = _size * 2
    VERTICES = []
    for z in range(-_size, _size+1):
        for a in range(-_size, _size+1):
            VERTICES.append(Vertex(a, -_size, z))
            VERTICES.append(Vertex(a, +_size, z))

            VERTICES.append(Vertex(-_size, a, z))
            VERTICES.append(Vertex(+_size, a, z))

    for y in range(-_size, _size+1):
        for x in range(-_size, _size+1):
            VERTICES.append(Vertex(x, y, -_size))
            VERTICES.append(Vertex(x, y, +_size))

    return Shape2D(Vertex(0, 0, 0), VERTICES, COLORS['g_cage'], GL_LINES)


def Cage(_size : int = 4):
    """
    Initialize 2D Cage (Borders) object .
    """
    VERTICES = [
        Vertex(
            -_size,
            -_size,
            -_size,
        ),
        Vertex(
            +_size,
            -_size,
            -_size,
        ),
        Vertex(
            -_size,
            -_size,
            -_size,
        ),
        Vertex(
            -_size,
            +_size,
            -_size,
        ),
        Vertex(
            -_size,
            -_size,
            -_size,
        ),
        Vertex(
            -_size,
            -_size,
            +_size,
        ),

        Vertex(
            -_size,
            +_size,
            -_size,
        ),
        Vertex(
            -_size,
            +_size,
            +_size,
        ),
        Vertex(
            -_size,
            +_size,
            -_size,
        ),
        Vertex(
            +_size,
            +_size,
            -_size,
        ),

        Vertex(
            +_size,
            +_size,
            +_size,
        ),
        Vertex(
            -_size,
            +_size,
            +_size,
        ),
        Vertex(
            +_size,
            +_size,
            +_size,
        ),
        Vertex(
            +_size,
            -_size,
            +_size,
        ),
        Vertex(
            +_size,
            +_size,
            +_size,
        ),
        Vertex(
            +_size,
            +_size,
            -_size,
        ),

        Vertex(
            +_size,
            -_size,
            +_size,
        ),
        Vertex(
            -_size,
            -_size,
            +_size,
        ),
        Vertex(
            +_size,
            -_size,
            +_size,
        ),
        Vertex(
            +_size,
            -_size,
            -_size,
        ),

        Vertex(
            -_size,
            -_size,
            +_size,
```

```python
        ),
        Vertex(
            -_size,
            +_size,
            +_size,
        ),
        Vertex(
            +_size,
            -_size,
            -_size,
        ),
        Vertex(
            +_size,
            +_size,
            -_size,
        ),

    ]

    return Shape2D(Vertex(0, 0, 0), VERTICES, COLORS['g_cage'], GL_LINES)


def Square(_position : Vertex = Vertex(0, 0, 0), _edge : float = 1, _color : tuple[float]
= None, _rotation : tuple[float, str] = None):
    """
    Initialize 2D Square object .
    """
    half_edge = _edge / 2
    VERTICES = [
        Vertex(
            _position.x - half_edge,
            _position.y - half_edge,
            _position.z,
        ),
        Vertex(
            _position.x + half_edge,
            _position.y - half_edge,
            _position.z,
        ),
        Vertex(
            _position.x + half_edge,
            _position.y + half_edge,
            _position.z,
        ),
        Vertex(
            _position.x - half_edge,
            _position.y + half_edge,
            _position.z,
        )
    ]
    square = Shape2D(_position, VERTICES, _color)
    if _rotation is not None:
        square.rotate(*_rotation)
    return square


def Cirlce(_position : Vertex = Vertex(0, 0, 0), _radius : float = 1, _n_sides : int =
100, _color : tuple[float] = None):
    """
    Initialize 2D Cirlce object .
    """
    angle = 2 * pi / _n_sides
    VERTICES = []
    for i in range(_n_sides):
        vertex = Vertex(
            _position.x,
            _position.y + _radius,
            _position.z,
        )
        vertex.rotate(_position, i * angle, 'Ro_z')
        VERTICES.append(vertex)
    return Shape2D(_position, VERTICES, _color)



```